Lecture 6: Probability Computations, Failure Events

Motivation

- some logic rules require proving lossless properties:
  e.g., one-sided tactic invocation on pRHL.
  (reminder: c is lossless if $\Pr[c : \mathit{true}] = 1$)
- reasoning on crypto proofs often involves bounding the probability of events.
  e.g., from reasoning on equivalence up to an event

Pr[f1 : res] $\leq$ Pr[f2: res] + Pr[f2 : event].

Bounded Hoare judgements

$$[c : \Psi \Longrightarrow \Phi] \leq \delta$$

- Ψ, Φ predicates on the initial and final memories (resp)
- δ a real expression evaluated in the inital memory

Interpretation

$$\forall m,\ \Psi\, m \Rightarrow [\![c]\!]\, m\, \mathbb{1}_{\Phi} \leq \delta\, m$$

Examples

```
module type Adv = {
  fun g():bool
}.

module M1 (A:Adv) = {
  var b1, b2:bool
  fun main () : unit = {
    b2 = A.g();
    b1 = ${0,1};
  }
}.

lemma ex10 (A<:Adv):
  bd_hoare[M1(A).main: true ⟹ M1.b2 = M1.b1] ≤ (1/2).
```

Examples

```
const qS : int.
axiom qS_pos : 0 < qS.

module type Adv2 = { fun h() : bitstring set }.

module M4(A : Adv2) = {
 var bs : bitstring set
 fun f () : bool ={
  var b : bitstring; var r : bool = false;
  bs = A.h();
  if (card bs < qS) { b = $dword; r = mem b bs;}
  return (r);
 }
}.

lemma ex14 (A <:Adv2) :
 islossless A.h ⇒
 bd_hoare[M4(A).f : true ⟹ res] ≤ (qS ∗ 1 / (2^l)).
```

Properties

relation to Pr expressions:

$$[c : \Psi \Longrightarrow \Phi] \leq \delta \quad \Longleftrightarrow \quad \forall m, \Psi\, m \Rightarrow \Pr[c, m : \Phi] \leq \delta$$

relation to standard Hoare Logic:

$$c : \Psi \Longrightarrow \Phi \quad \Longleftrightarrow \quad [c : \Psi \Longrightarrow \neg\Phi] = 0$$

$$c : \Psi \Longrightarrow \Phi \wedge [c : \Psi \Longrightarrow \mathsf{true}] = \delta \quad \Longleftrightarrow \quad [c : \Psi \Longrightarrow \Phi] = \delta$$

Skip statements

$$\frac{\Psi \Rightarrow \Phi \qquad \delta = 1}{[\text{skip} : \Psi \Longrightarrow \Phi] = \delta}$$

- why 1?
- what if = 0? when post doesn't hold?
  In that case one can rely on this property:

$$[c : P \Longrightarrow Q] = 0 \Leftrightarrow c : P \Longrightarrow \neg Q$$

Other trivial tactics

exfalso:

$$\frac{}{[c : \mathit{false} \Longrightarrow \Phi] \leq \delta}$$

pr_bounded:

$$\frac{}{[c : \Psi \Longrightarrow \Phi] \leq 1} \qquad \frac{}{[c : \Psi \Longrightarrow \Phi] \geq 0}$$

Other trivial tactics

exfalso:

$$\frac{}{[c : \mathit{false} \Longrightarrow \Phi] \leq \delta}$$

pr_bounded:

$$\frac{}{[c : \Psi \Longrightarrow \Phi] \leq 1} \qquad \frac{}{[c : \Psi \Longrightarrow \Phi] \geq 0}$$

both, among others, components of the “trivial” tactic for Hoare judgements.

Reasoning about sequential composition

Main difficulty

$$[c_1 ; c_2 : \Psi \Longrightarrow \Phi] = \delta$$

(what to claim about intermediate program point?)

If c_1 or c_2 are deterministic statements then it is often simple.

But does a rule like the following hold?

$$\frac{[c_1 : \Psi \Longrightarrow \chi] = \delta_1 \qquad [c_2 : \chi \Longrightarrow \Phi] = \delta_2}{[c_1 ; c_2 : \Psi \Longrightarrow \Phi] = \delta_1 \delta_2}$$

Main difficulty

The answer is NO.

$$[x = \$\{0,1\}; y = \$\{0,1\} : \mathit{true} \Longrightarrow x \vee y] = \frac{3}{4}$$

$$\frac{[x = \$\{0,1\} : \mathit{true} \Longrightarrow x] = \frac{1}{2} \qquad [y = \$\{0,1\} : x \Longrightarrow x \vee y] = 1}{[x = \$\{0,1\}; y = \$\{0,1\} : \mathit{true} \Longrightarrow x \vee y] = ??}$$

seq tactic

Syntax: seq $\chi\ \delta_1\ \delta_2\ \delta_3\ \delta_4$

$$\frac{\begin{array}{cc}[c_1 : \Psi \Longrightarrow \chi] \leq \delta_1 & [c_2 : \chi \Longrightarrow \Phi] \leq \delta_2 \\ [c_1 : \Psi \Longrightarrow \neg\chi] \leq \delta_3 & [c_2 : \neg\chi \Longrightarrow \Phi] \leq \delta_4 \\ \multicolumn{2}{c}{\delta_1\delta_2 + \delta_3\delta_4 \leq \delta}\end{array}}{[c_1; c_2 : \Psi \Longrightarrow \Phi] \leq \delta}$$

- case analysis on intermediate program point
- bound splitting

seq tactic

example:

$$[x = \$\{0,1\}; y = \$\{0,1\} : \mathit{true} \Longrightarrow x \vee y] = \frac{3}{4}$$

verification subgoals:

$$\frac{\begin{array}{cc}[x = \$\{0,1\} : \mathit{true} \Longrightarrow x] = \frac{1}{2} & [y = \$\{0,1\} : x \Longrightarrow x \vee y] = 1 \\ [x = \$\{0,1\} : \mathit{true} \Longrightarrow \neg x] = \frac{1}{2} & [y = \$\{0,1\} : \neg x \Longrightarrow x \vee y] = \frac{1}{2}\end{array}}{[x = \$\{0,1\}; y = \$\{0,1\} : \mathit{true} \Longrightarrow x \vee y] = \frac{1}{2} + \frac{1}{2}\frac{1}{2}}$$

seq tactic

Fortunately, the rule application can be often simplified.

E.g., if c_1 is deterministic:

$$\frac{\begin{array}{cc} [c_1 : \Psi \Longrightarrow \chi] = \delta_1 (= 1) & [c_2 : \chi \Longrightarrow \Phi] = \delta_2 (= \delta) \\ [c_1 : \Psi \Longrightarrow \neg\chi] = \delta_3 (= 0) & [c_2 : \neg\chi \Longrightarrow \Phi] = \delta_4 (= ?) \\ \multicolumn{2}{c}{\delta_1 \delta_2 + \delta_3 \delta_4 \leq \delta} \end{array}}{[c_1; c_2 : \Psi \Longrightarrow \Phi] = \delta}$$

seq χ $\not{\delta}_1$ $\not{\delta}_2$ $\not{\delta}_3$ $\not{\delta}_4$

seq tactic

However, still some limitations...

```
{true}
x = [1..10];
y = [1..10];
{x < y} <= 9/20
```

What would you suggest as intermediate assertion?

seq tactic

However, still some limitations...

```
{true}
x = [1..10];
y = [1..10];
{x < y} <= 9/20
```

What would you suggest as intermediate assertion?

Is there candidate for the rule generalization?

wp tactic

deterministic straight-line code

- assignments, and
- conditionals contaning only straight-line code

$$\frac{[c_1 : \Psi \Longrightarrow \mathrm{wp}(c_2, \Phi)] = \delta}{[c_1; c_2 : \Psi \Longrightarrow \Phi] = \delta}$$

- bound expression δ is not modified, depends exclusively of the initial memory,
- no need to reason on intermediate case analysis, nor bound splitting:

$$\frac{\begin{array}{cc} [c_1 : \Psi \Longrightarrow \chi] = \delta_1 (= \delta) & [c_2 : \chi \Longrightarrow \Phi] = \delta_2 (= 1) \\ [c_1 : \Psi \Longrightarrow \neg\chi] = \delta_3 (= ?) & [c_2 : \neg\chi \Longrightarrow \Phi] = \delta_4 (= 0) \end{array}}{[c_1; c_2 : \Psi \Longrightarrow \Phi] = \delta_1\delta_2 + \delta_3\delta_4}$$

wp tactic

deterministic straight-line code

- assignments, and
- conditionals contaning only straight-line code

$$\frac{[c_1 : \Psi \implies \mathrm{wp}(c_2, \Phi)] = \delta}{[c_1; c_2 : \Psi \implies \Phi] = \delta}$$

- bound expression δ is not modified, depends exclusively of the initial memory,
- no need to reason on intermediate case analysis, nor bound splitting:

$$\frac{\begin{array}{cc} [c_1 : \Psi \implies \chi] = \delta_1 (= \delta) & [c_2 : \chi \implies \Phi] = \delta_2 (= 1) \\ [c_1 : \Psi \implies \neg\chi] = \delta_3 (= ?) & [c_2 : \neg\chi \implies \Phi] = \delta_4 (= 0) \end{array}}{[c_1; c_2 : \Psi \implies \Phi] = \delta_1\delta_2 + \delta_3\delta_4}$$

wp tactic

deterministic straight-line code

- assignments, and
- conditionals contaning only straight-line code

$$\frac{[c_1 : \Psi \Longrightarrow \mathrm{wp}(c_2, \Phi)] = \delta}{[c_1 ; c_2 : \Psi \Longrightarrow \Phi] = \delta}$$

- bound expression δ is not modified, depends exclusively of the initial memory,
- no need to reason on intermediate case analysis, nor bound splitting:

$$\frac{\begin{array}{cc} [c_1 : \Psi \Longrightarrow \chi] = \delta_1 (= \delta) & [c_2 : \chi \Longrightarrow \Phi] = \delta_2 (= 1) \\ [c_1 : \Psi \Longrightarrow \neg\chi] = \delta_3 (= ?) & [c_2 : \neg\chi \Longrightarrow \Phi] = \delta_4 (= 0) \end{array}}{[c_1 ; c_2 : \Psi \Longrightarrow \Phi] = \delta_1 \delta_2 + \delta_3 \delta_4}$$

wp example

```
module M = {
  var x : real
  var y : bool
  fun foo () : unit = {
    if (y) {
      y = ${0,1};
    }
    x = 1%r;
  }
}
```

$$\left[\text{if } b \text{ then } y = \$\{0,1\}; x = 1 : M.x = \tfrac{1}{2} \vee M.y \implies M.y \vee M.x = 1\right] = M.x$$

wp (+simplify)

$$\left[\text{if } b \text{ then } y = \$\{0,1\} : M.x = \tfrac{1}{2} \vee M.y \implies M.y\right] = M.x$$

Distr library

some reminder on the formalization of distributions:

- α *distr* primite type
- μ : α *distr* $\rightarrow$ ($\alpha \rightarrow$ *bool*) $\rightarrow$ real
- **op** in_supp x (d:α *distr*) : bool = 0%r < μ_x d x
- **op** μ_x(d:α *distr*, x) : real = μ d ((=) x)
- **op** weight(d:α *distr*) : real = μ d cpTrue

rnd tactic (general version)

Syntax: *rnd* $\varphi\ \delta_1\ \delta_2\ \delta_3\ \delta_4$

$$\frac{\begin{array}{c}\delta_1\delta_2 + \delta_3\delta_4 \leq \delta \\ [c : \Psi \Longrightarrow \varphi] \leq \delta_1 \\ \varphi \Rightarrow \mu\, d\, (\lambda v, v \in \mathrm{supp}\, d \Rightarrow \Phi\, [^v/_x]) \leq \delta_2 \\ [c : \Psi \Longrightarrow \neg\varphi] \leq \delta_3 \\ \neg\varphi \Rightarrow \mu\, d\, (\lambda v, v \in \mathrm{supp}\, d \Rightarrow \Phi\, [^v/_x]) \leq \delta_4\end{array}}{[c; x = \$d : \Psi \Longrightarrow \Phi] \leq \delta}$$

rnd tactic (general version)

Syntax: *rnd* $\varphi\ \delta_1\ \delta_2\ \delta_3\ \delta_4\ p$

$$\frac{\begin{array}{c}\delta_1\delta_2 + \delta_3\delta_4 \leq \delta \\ [c : \Psi \Longrightarrow \varphi] \leq \delta_1 \\ \varphi \Rightarrow \mu\, d\, p \leq \delta_2 \wedge (\forall v,\ v \in \mathrm{supp}\, d \Rightarrow \Phi\, [^v/_x] \Rightarrow p\, v) \\ [c : \Psi \Longrightarrow \neg\varphi] \leq \delta_3 \\ \neg\varphi \Rightarrow \mu\, d\, p \leq \delta_4 \wedge (\forall v,\ v \in \mathrm{supp}\, d \Rightarrow \Phi\, [^v/_x] \Rightarrow p\, v)\end{array}}{[c; x = \$d : \Psi \Longrightarrow \Phi] \leq \delta}$$

It accepts an extra parameter, equivalent to postcondition Φ

rnd tactic example

$$[x = \$\{0,1\}; y = \$\{0,1\} : \mathit{true} \Longrightarrow x \vee y] = \frac{3}{4}$$

$$\frac{\tfrac{1}{2} + \tfrac{1}{2}\tfrac{1}{2} \leq \tfrac{3}{4} \quad [x = \$\{0,1\} : \mathit{true} \Longrightarrow x] \leq \tfrac{1}{2} \quad x \Rightarrow \mu\,\{0,1\}\,(\lambda z.\, x \vee z) \leq 1 \quad [x = \$\{0,1\} : \mathit{true} \Longrightarrow \neg x] \leq \tfrac{1}{2} \quad \neg x \Rightarrow \mu\,\{0,1\}\,(\lambda z.\, x \vee z) \leq \tfrac{1}{2}}{[x = \$\{0,1\}; y = \$\{0,1\} : \mathit{true} \Longrightarrow x \vee y] \leq \tfrac{3}{4}}$$

simplified rnd tactic (upper bounded)

Syntax: rnd

x occurs in postcondition Φ:

$$\frac{c : \Psi \Longrightarrow \mu\, d\, \Phi \leq \delta}{[c; x = \$d : \Psi \Longrightarrow \Phi] \leq \delta}$$

simplified rnd tactic (upper bounded)

Syntax: rnd

x occurs in postcondition Φ:

$$\frac{c : \Psi \Longrightarrow \mu\, d\, \Phi \leq \delta}{[c; x = \$d : \Psi \Longrightarrow \Phi] \leq \delta}$$

x does not occur in postcondition Ψ:

$$\frac{[c : \Psi \Longrightarrow \Phi] \leq \delta}{[c; x = \$d : \Psi \Longrightarrow \Phi] \leq \delta}$$

simplified rnd tactic (lower bounded)

Syntax: rnd

x occurs in postcondition:

$$\frac{[c : \Psi \Longrightarrow \mu\, d\, \Phi \geq \delta] = 1}{[c; x = \$d : \Psi \Longrightarrow \Phi] \geq \delta}$$

simplified rnd tactic (lower bounded)

Syntax: rnd

x occurs in postcondition:

$$\frac{[c : \Psi \Longrightarrow \mu\, d\, \Phi \geq \delta] = 1}{[c; x = \$d : \Psi \Longrightarrow \Phi] \geq \delta}$$

x does not occur in postcondition:

$$\frac{[c : \Psi \Longrightarrow \Phi] \geq \delta \qquad \mu\, d\, cpTrue = 1}{[c; x = \$d : \Psi \Longrightarrow \Phi] \geq \delta}$$

simplified rnd tactic (lower bounded)

Syntax: rnd

x occurs in postcondition:

$$\frac{[c : \Psi \Longrightarrow \mu\, d\, \Phi \geq \delta] = 1}{[c; x = \$d : \Psi \Longrightarrow \Phi] \geq \delta}$$

x does not occur in postcondition:

$$\frac{[c : \Psi \Longrightarrow \Phi] \geq \delta \qquad \mu\, d\, cpTrue = 1}{[c; x = \$d : \Psi \Longrightarrow \Phi] \geq \delta}$$

As with the most general tactic variant we can add an optional postcondition argument. What for?

rnd tactic examples

```
module M = {
  var b1, b2 : bool
  fun f () : unit = {
    b1 = ${0,1};
    b2 = ${0,1};
  }
}.

lemma ex1 : bd_hoare[M.f : true ==> M.b2] = (1/2).

lemma ex2 : bd_hoare[M.f : true ==> M.b1] = (1/2).
```

if tactic

$$\frac{[c_1; c : \Psi \wedge b \Longrightarrow \Phi] \leq \delta \qquad [c_2; c : \Psi \wedge \neg b \Longrightarrow \Phi] \leq \delta}{[\text{if } b \text{ then } c_1 \text{ else } c_2; c : \Psi \Longrightarrow \Phi] \leq \delta}\ [\textbf{if}]$$

```
module M2 = {
  var b,b' : bool
  fun f () : unit = {
    if (b) {
      b' = false;
    } else {
      b' = $ {0,1}\(single b);
    }
  }
}.
lemma test : bd_hoare [M2.f : true ==> M2.b \/ M2.b' ] = 1.
```

call tactic

Syntax: call Ψ_f Φ_f

$$\frac{c : \Psi \Longrightarrow \Psi_f \left[{}^{\vec{y}}/_{\vec{p}}\right] \wedge \forall\, v\, \vec{z}.\ \Phi_f \left[{}^{v}/_{\mathrm{res}_f}\right] \left[{}^{\vec{z}}/_{\vec{m}}\right] \Rightarrow \Phi \left[{}^{v}/_{x}\right] \left[{}^{\vec{z}}/_{\vec{m}}\right] \qquad [f : \Psi_f \Longrightarrow \Phi_f] \leq \delta}{\left[c; x = f(\vec{y}) : \Psi \Longrightarrow \Phi\right] \leq \delta}$$

$$\frac{\left[c : \Psi \Longrightarrow \Psi_f \left[{}^{\vec{y}}/_{\vec{p}}\right] \wedge \forall\, v\, \vec{z}.\ \Phi_f \left[{}^{v}/_{\mathrm{res}_f}\right] \left[{}^{\vec{z}}/_{\vec{m}}\right] \Rightarrow \Phi \left[{}^{v}/_{x}\right] \left[{}^{\vec{z}}/_{\vec{m}}\right]\right] = 1 \qquad [f : \Psi_f \Longrightarrow \Phi_f] \geq \delta}{\left[c; x = f(\vec{y}) : \Psi \Longrightarrow \Phi\right] \geq \delta}$$

while tactic

Syntax: while χ e

$$\frac{\begin{array}{c}[c' : \Psi \Longrightarrow \chi \wedge \forall M.\ (\chi \wedge 0 \leq e \Rightarrow \neg b) \wedge (\chi \wedge \neg b \Rightarrow \Phi)] \leq \delta \\ \forall k.\ [c : \chi \wedge b \wedge e = k \Longrightarrow \chi \wedge e < k] = 1\end{array}}{[c'; \text{while } b \text{ do } c : \Psi \Longrightarrow \Phi] \leq \delta}$$

Failure event lemma

```
module O : O = {
  var bad : bool; var m : (from, to) map; var s : to list;
  fun init() : unit = { bad = false; m = Map.empty; s = []; }
  fun o(x:from) : to = {
    if (length s < qO ) {
      y = $dsample; if (List.mem y s) bad = true;
      if (!in_dom x m) m[x] = y; s = y :: s;
    }
    return (proj (m[x]));
  }
}
module M(A:Adv) = {
  module AO = A(O)
  fun main () : unit = { O.init(); AO.g(); }
}
lemma test : ∀(A<:Adv{O}), ∀&m,
Pr[M(A).main() @ &m : O.bad] ≤ qO ∗ (qO−1) ∗ bd.
```

Failure event lemma

fel n q h c F P

- c_1, c_2 stands for the splitting of its body at position n
- $\{O_i\}_{i=0}^k$: all oracles accessed by any adversary called at c_2
- variables in F can only be modified by c_1 and $\{O_i\}_{i=0}^k$

$$\left\{\begin{array}{l} [O_i : \neg F \Longrightarrow F] \leq h(c) \\ \forall c_0,\ O_i : P \wedge c = c_0 \Longrightarrow c_0 < c \\ \forall c_0,\ \forall f_0,\ O_i : \neg P \wedge F = f_0 \wedge c = c_0 \Longrightarrow F = f_0 \wedge c = c_0 \end{array}\right\}_{i=0}^{k}$$

$$\forall m', (\varphi \Rightarrow F \wedge c \leq q) \qquad \sum_{i=0}^{q-1} h(i) \leq \epsilon$$

$$c_1 : \text{true} \Longrightarrow \neg F \wedge c = 0$$

$$\overline{\Pr[f, m : \varphi] \leq \epsilon}$$

FEL example

Bound and counter initialisation goal (+inlining):

```
pre = true

    O.bad = false; O.m = Map.empty; O.s = [];

post = ! O.bad /\ length O.s = 0
```

FEL example

Goal on probability of setting bad:

```
pre = 0 <= length O.s /\ ! O.bad

      if (length O.s < qO) {
         y = $dsample;
         if (mem y O.s) { O.bad = true }
         if (! in_dom x O.m) { O.m[x] = y }
         O.s = y :: O.s
      }
      r = proj O.m.[x]

post = O.bad

Bound : [<=] (length O.s)%r * bd
```